Apex Analyst - Racing Driver Performance Analysis Tool

Overview

Apex Analyst is a comprehensive, data-driven platform designed to help amateur and semi-professional racing drivers identify areas for improvement, optimize their racing lines, and better understand performance through objective telemetry analysis.

Problem Statement

Amateur and semi-professional racing drivers struggle to effectively identify specific areas for improvement due to limited access to comprehensive, data-driven analysis. Current methods rely on subjective "feel" or basic metrics that don't pinpoint root causes like suboptimal braking points, gear selection, or cornering lines.

Features

Core Analysis Modules

1. **Racing Line Optimization**
   - Visual overlay of driver's line vs. optimal reference line
   - Highlights deviations in corner entry, apex, and exit points
   - Identifies problem corners with severity ratings
2. **Braking & Acceleration Strategy**
   - Analyzes braking points, pressure, and zones
   - Compares G-force ranges to benchmark
   - Identifies early/late braking and pressure issues
3. **Gear Selection Analysis**
   - Flags incorrect gear selections
   - Suggests optimal gears for power delivery
   - Detects gear hunting patterns
4. **Performance Scoring & Profiling**
   - Machine learning-based overall performance score
   - Driver behavioral profiling
   - Identifies tendencies (cautious, aggressive, etc.)

Output & Insights

- **Prioritized Improvement Areas**: Top 3 areas with most significant potential
- **Interactive Visualizations**: Side-by-side data comparisons

- **Tailored Drills**: Specific practice recommendations
- **Comprehensive Reports**: Text and visual analysis summaries

Project Structure

```
apex_analyst/
├── apex_analyst.py          # Main application entry point
├── data_acquisition.py      # Telemetry data simulation and management
├── racing_line_analyzer.py  # Racing line analysis module
├── braking_analyzer.py      # Braking performance analysis
├── gear_analyzer.py         # Gear selection analysis
├── performance_scorer.py    # Performance scoring and profiling
├── report_generator.py      # Report and visualization generation
├── requirements.txt         # Python dependencies
├── README.md                # This file
└── output/                  # Generated reports and visualizations
    ├── racing_line_comparison.png
    ├── speed_trace.png
    ├── braking_analysis.png
    ├── gear_usage.png
    ├── performance_dashboard.png
    └── analysis_report.txt
```

Installation

Prerequisites

- Python 3.8 or higher
- pip package manager

Setup

1. Clone or download the project:

```bash
git clone <repository-url>
cd apex_analyst
```

2. Install dependencies:

```bash
pip install -r requirements.txt
```

Usage

Basic Usage

Run the main analysis:

```bash
python apex_analyst.py
```

This will:

1. Generate simulated telemetry data (driver + reference)
2. Perform comprehensive analysis
3. Generate visualizations and reports in the `output/` folder
4. Display prioritized improvement recommendations

Custom Analysis

```python
from apex_analyst import ApexAnalyst

# Initialize analyzer
analyst = ApexAnalyst(track_name="Silverstone GP")

# Load session data
analyst.load_session_data(driver_name="Reference Driver", is_reference=True)
analyst.load_session_data(driver_name="Your Name", is_reference=False)

# Run analyses
analyst.analyze_racing_line()
analyst.analyze_braking()
analyst.analyze_gears()
analyst.calculate_performance_score()

# Generate report
analyst.generate_report()

# Get improvement plan
analyst.display_improvement_plan()
```

Working with Real Data

To use real telemetry data instead of simulated data, modify the `data_acquisition.py` module to load from your data logger format (CSV, JSON, etc.):

```python
def load_real_data(filepath):
    """Load real telemetry data from file"""
    # Implement your data loading logic
    # Return dictionary with keys: distance, time, speed, throttle, brake, gear, lat_g, long_g, x_position, y_position
    pass
```

Output Examples

1. Performance Dashboard

- Overall performance score (0-100)
- Component scores breakdown
- Driver profile and characteristics
- Improvement potential estimation

2. Racing Line Comparison

- Visual track map overlay
- Color-coded deviation indicators
- Problem corner highlights

3. Speed Trace

- Speed comparison throughout lap
- Areas of speed deficit highlighted
- Corner-by-corner analysis

4. Braking Analysis

- Brake pressure comparison
- Braking point differences per turn
- Early/late braking identification

5. Gear Usage

- Gear selection comparison
- Mismatch highlighting
- Optimal gear recommendations

6. Text Report

- Comprehensive analysis summary

- Detailed metrics and statistics
- Prioritized improvement recommendations

Performance Scoring System

Component Scores (0-100)

- **Racing Line** (35% weight): Measures track position accuracy
- **Braking** (30% weight): Evaluates braking timing and pressure
- **Gear Selection** (20% weight): Assesses gear choice efficiency
- **Consistency** (15% weight): Measures performance variation

Driver Profiles

The system identifies driver behavioral patterns:

- **Cautious Conservative**: Early braking, wider lines
- **Smooth Operator**: Excellent all-around performance
- **Developing Amateur**: Multiple improvement areas
- **Technical Learner**: Good fundamentals, gear focus needed
- **Aggressive Pusher**: Late braking, confident style
- **Balanced Intermediate**: Solid all-around performance

Customization

Track Configuration

Modify track corners in `data_acquisition.py`:

```python
self.corners = [
    {'name': 'Turn 1', 'distance': 450, 'radius': 80, 'ideal_speed': 180, 'ideal_gear': 5},
    # Add more corners...
]
```

Scoring Weights

Adjust component weights in `performance_scorer.py`:

```python
```

```
weights = {
    'racing_line': 0.35,
    'braking': 0.30,
    'gear_selection': 0.20,
    'consistency': 0.15
}
```

Future Enhancements

- [ ] Multi-lap consistency analysis
- [ ] Real-time telemetry integration
- [ ] Machine learning for predictive recommendations
- [ ] Video overlay synchronization
- [ ] Comparative analysis (multiple drivers)
- [ ] Mobile app integration
- [ ] Cloud-based data storage
- [ ] Social sharing features
- [ ] Virtual coaching assistant

Technical Details

Data Structure

Telemetry data includes:

- **Distance**: Track position (meters)
- **Time**: Elapsed time (seconds)
- **Speed**: Vehicle speed (km/h)
- **Throttle**: Throttle position (0-100%)
- **Brake**: Brake pressure (0-100%)
- **Gear**: Current gear (1-6)
- **Lateral G**: Lateral acceleration (G's)
- **Longitudinal G**: Forward/braking acceleration (G's)
- **Position**: X/Y coordinates on track

Analysis Algorithms

1. **Line Deviation**: Euclidean distance calculation
2. **Braking Detection**: Threshold-based zone identification
3. **Gear Efficiency**: Speed-range matching algorithm

4. **Performance Scoring**: Weighted component aggregation

Contributing

Contributions are welcome! Areas for contribution:

- Real data logger integrations
- Additional analysis modules
- Visualization improvements
- Machine learning models
- Documentation enhancements

License

MIT License - Feel free to use and modify for your racing analysis needs.

Support

For issues, questions, or feature requests, please open an issue on the project repository.

Acknowledgments

Built to address the performance analysis gap in amateur and semi-professional motorsports, enabling data-driven improvement and safer, faster driving.

Remember: The goal is focused, efficient skill development through objective, measurable feedback. Safe and fast!